Problem 8 (20 points)

Problem Description

In this problem you will use `sklearn.svm.SVC` to classify thermal imaging data of a CPU die. We are interested in classifying points on the die as critical or non-critical, to inform where thermal paste should be applied to the die. The thermal imaging data is noisy, so your boss has asked you to develop a model that can produce a smoother profile of where the die is expected to be at, or above critical temperature.

The thermal imaging data is contained in `cputemp.npy`, where the first two columns correspond to the x and y position on the die, and the third column corresponds to the temperature at that point in degrees Celsius.

Fill out the notebook as instructed, making the requested plots and printing necessary values.

You are welcome to use any of the code provided in the lecture activities.

Summary of deliverables:

Functions:

- `accuracy(model, X, y)`

Results:

- Print the accuracy of the two models requested on classifying the training set points as critical or non-critical temperature

Plots:

- Plot the decision boundary of each trained model with the provided plotting functions

Discussion:

- Compare the plots and accuracy of the two models, and reason which model is the better of the two

Imports and Utility Functions:

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.svm import SVC

def plot_svc_decision_function(model, ax=None):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
```

```
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 50)
    y = np.linspace(ylim[0], ylim[1], 50)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary and margins
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1],
               linestyles=['--', '-', '--'],
               linewidths = [2,4,2])

    ax.set_xlim(xlim)
    ax.set_ylim(ylim)
    plt.show()

def plot_temp_profile(X, T, ax = None):
    if ax == None:
        ax = plt.gca()
    # Plot points colored by temperature
    sc = ax.scatter(X[:,0],X[:,1],c = T)
    # Add colorbar to plot
    cbar = plt.colorbar(sc)
    # Add labels
    cbar.set_label('Temperature ($\degree C$)')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    plt.show()

def plot_temp_critical(X, y, ax = None):
    if ax is None:
        ax = plt.gca()
        showflag = True
    else:
        showflag = False
    ax.scatter(X[:,0],X[:,1], c = y, cmap = ListedColormap(['blue', 'red']))
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_aspect(0.8)
    if showflag:
        plt.show()
    else:
        return ax

def plot_model(model, X, y):
    # Wrapper function to generate plot and decision boundary
    ax = plt.gca()
    ax = plot_temp_critical(X,y,ax)
    plot_svc_decision_function(model, ax)
```

Load and visualize the data

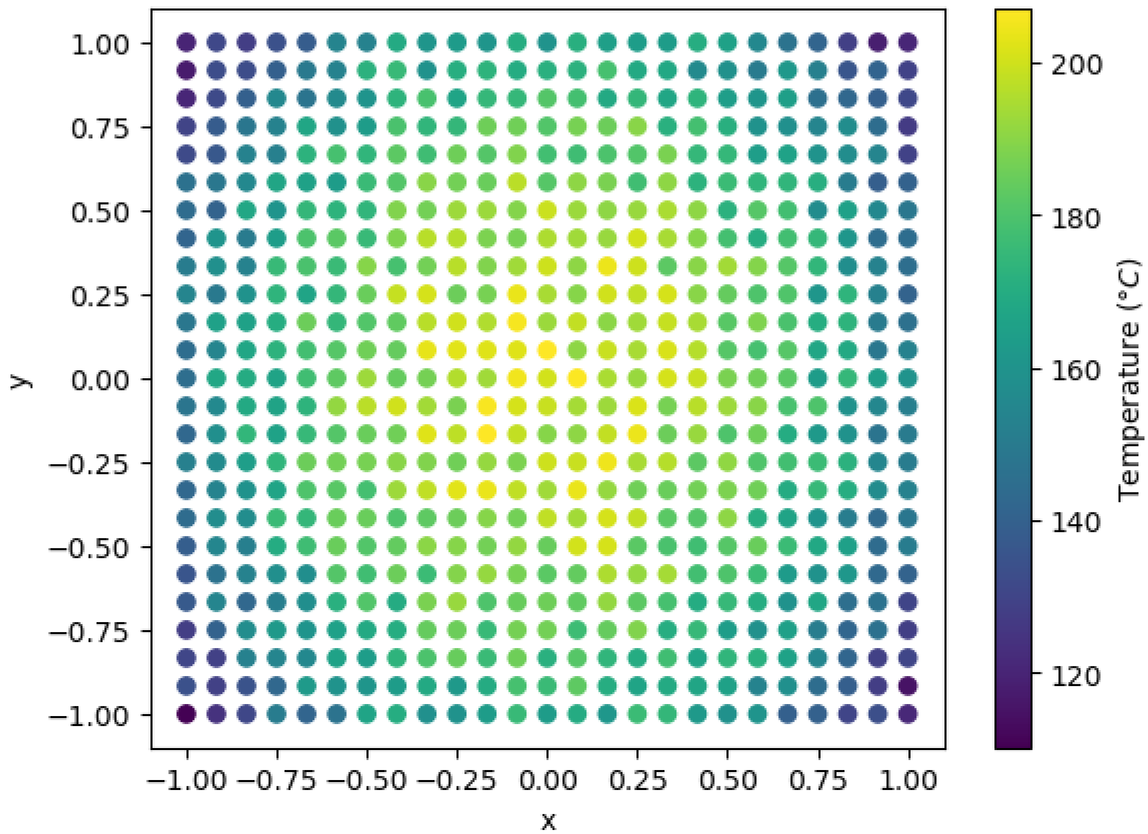
Data is contained in `cputemp.npy` and can be loaded with `np.load()`. The first two columns of the file correspond to the x and y position on the die, and the third columns corresponds to the temperature at that position in degrees Celsius.

Store the data as:

- `X` (Nx2) array of position data
- `T` (Nx1) array of temperature data

Then visualize the data with `plot_temp_profile(X,T)`

In [9]:
```
# YOUR CODE GOES HERE
cputemp = np.load("data/cputemp.npy")
X = np.array([cputemp[:,0],cputemp[:,1]]).T
T = cputemp[:,2]

plot_temp_profile(X,T)
```

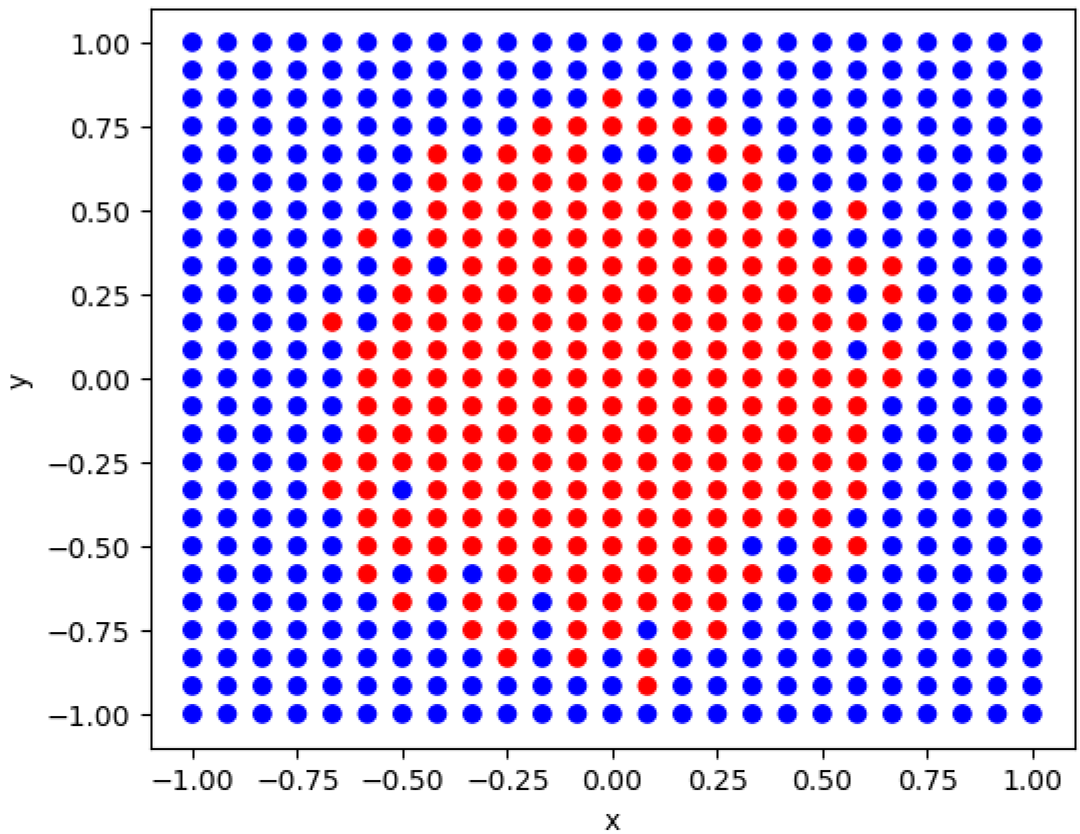


Assign labels to data

Now we need to assign labels to the data for the support vector machine to be able to classify points as critical or non-critical. Generate a boolean vector `y` that is True for points at or above $180 \text{\textdegree}C$, and False otherwise. Then use `plot_temp_critical(X,y)` to plot the points on the die that are critical and non-critical.

In [10]:
```
# YOUR CODE GOES HERE
y = np.zeros(len(T),dtype = bool)
```

```
for i in range(len(T)):
    if (T[i] > 180):
        y[i] = True
    else:
        y[i] = False

plot_temp_critical(X,y)
```



Train Support Vector Classifiers

Now you can train a SVC to classify the region on the die that you expect to be at or above the critical temperature. Using sklearn.svm.SVC train the following two models:

- RBF Kernel with C = 100
- 8th order polynomial Kernel with C = 100

Write a function `accuracy(model, X, y)` that takes in the model, evaluates the points in X, and computes an accuracy between the predictions and ground truth labels in `y`. Accuracy is defined as the number of correctly classified points, divided by the total number of points. For a more in depth discussion of accuracy please see: Accuracy - Wikipedia. We will cover this topic more later in the course.

For each model, report the accuracy on the training data and use `plot_model(model, X, y)` to visualize the decision boundary.

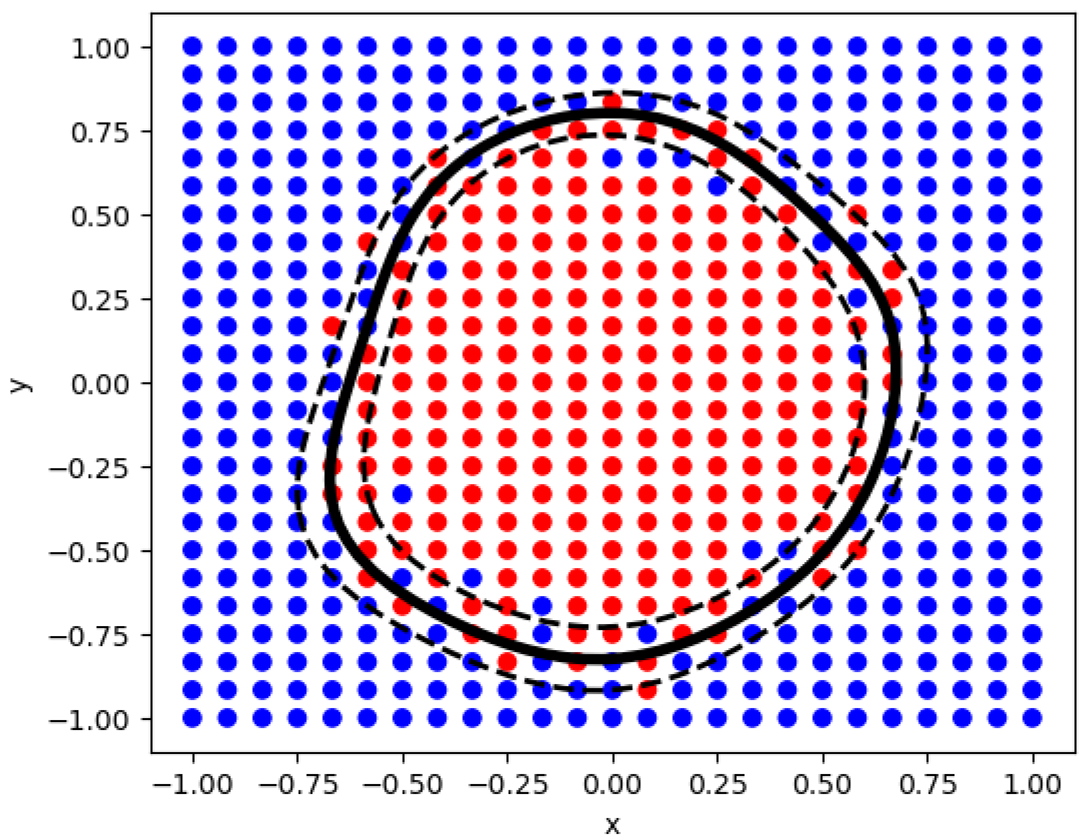
In [11]:
```
# YOUR CODE GOES HERE
# Define accuracy function
```

```python
def accuracy(model,X,y):
    pred = model.predict(X)
    acc = np.sum(pred == y) / len(y) * 100
    return acc
```

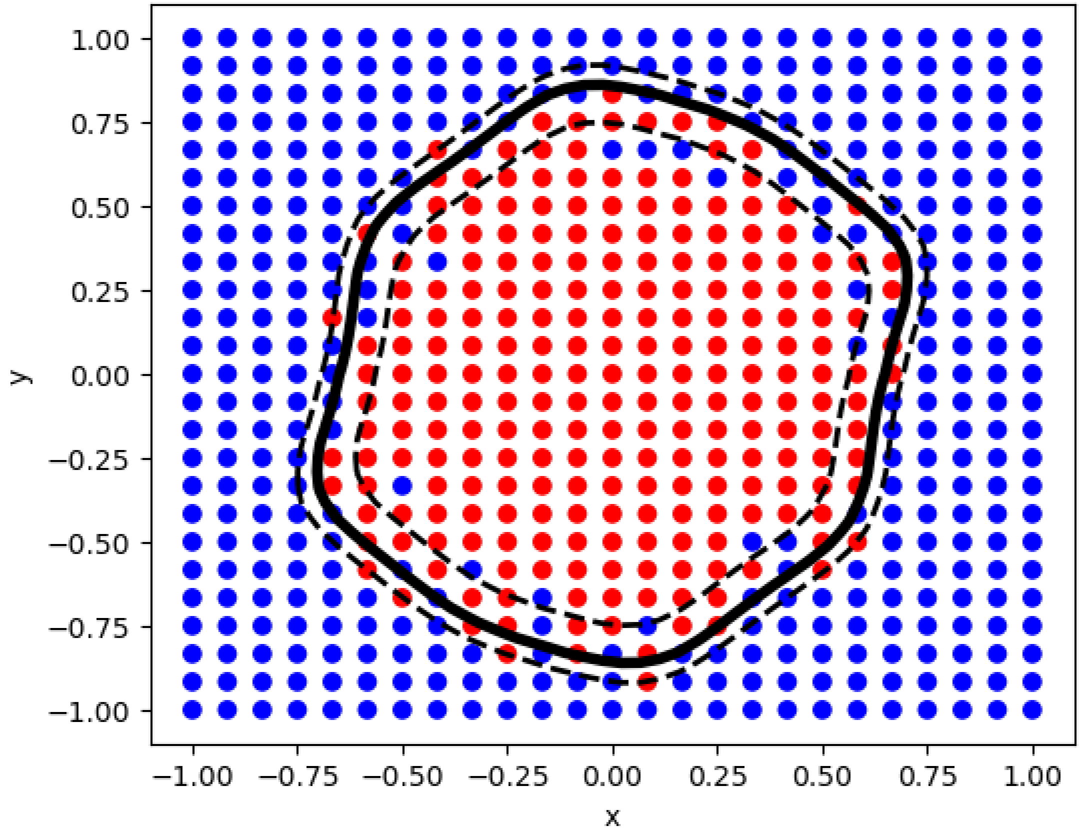
In [12]:

```python
# YOUR CODE GOES HERE
# Train and plot SVC models
C=100
model_1 = SVC(kernel="rbf",C=100)
model_1.fit(X,y)
accuracy_rbf = accuracy(model_1,X,y)

model_2 = SVC(kernel="poly",degree=8,C=100)
model_2.fit(X,y)
accuracy_poly = accuracy(model_2,X,y)

print("Accuracy for rbf model is: ",accuracy_rbf,"%")
print("Accuracy for 8th order polynomial model is: ",accuracy_poly,"%")

print("\n rbf model")
plot_model(model_1,X,y)

print("8th order polynomial model")
plot_model(model_2,X,y)
```

```
Accuracy for rbf model is:  93.28 %
Accuracy for 8th order polynomial model is:  92.32000000000001 %

 rbf model
```




```
8th order polynomial model
```



Discussion

Briefly discuss the performance of the two models, both with regard to their accuracy and the appearance of the decision boundary. Which model would you submit to your boss?

The two models considered are rbf model and 8th order polynomial model. The rbf model has an accuracy of 93.28% and the 8th order polynomial has an accuracy of 92.32%. Based on the accuracy, the rbf is the better model as it provides more accurate answers and better results. Based on the plots, the rbf model provides a better decision boundary as it is more defined and smoother as compared to the polynomial. The decision boundary is considered to be smooth and defined and the kernel doesn't overfit.

The model I would submit to the boss is the rbf model.

```
In [ ]:
```